A Review of AI-based Resource Allocation Approaches in Cloud Environments

Li Zhong
li.zhong@student.uva.nl
13509306

March 3, 2022

Abstract

Cloud computing, as a pay-as-you-go model, has become a popular choice in science and industry because of its elasticity and simplicity. An efficient resource allocation mechanism is necessary to completely lease the elasticity while minimizing the resources cost and meeting the Service Level Agreement (SLA). In this paper, we study the application of Artificial Intelligence (AI) in resource allocation for cloud environments. This research is divided into two categories: resource provisioning in serverful clouds and serverless clouds. Serverless computing is a new execution model of cloud computing, in which the cloud provider takes care of the servers and resources on behalf of the cloud customers and releases the workload of users. Considering its difference in execution and popularity recently, we will discuss it individually. In particular, we survey the common challenges of resource allocation in clouds, and we investigate and compare AI-based techniques in auto-scaling in normal clouds and serverless clouds.

1 Introduction

Cloud computing, as an emerging technology that delivers different kinds of services, such as data storage, computing, and networking, is becoming increasingly popular. Cloud computing allows users to deploy their applications elastically while ensuring the efficiency of the infrastructure and charges only for resources actually consumed by users to save the total cost. There are three traditional business models in cloud computing, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [1]. In recent times, a new paradigm appears in cloud computing, which is serverless cloud computing or Function-as-a-Service (FaaS). Serverless does not mean there is no server in the system, but it refers to the transfer of jobs about deploying and managing resources on servers from application developers to service providers. Serverless cloud computing reduces the workload of users and becomes more and more popular.

As a pay-as-you-go model, a well-performed resource allocation mechanism is necessary for the cloud to meet the service-level agreement (SLA) while minimizing user cost and achieving the elasticity of clouds. To make full use of the elasticity, it's important to automate the whole process of resource provisioning without humans' intervention. The need of automating the process motivates a lot of research on the field of auto-scaling, which aims to find effective approaches to dynamically allocate resources more cost-efficiently without violating the SLA.

Because of the recent succcess of Artificial Intelligence (AI) in research problems in science and industry, AI is getting increasing attention and appears in research about auto-scaling in clouds. Commonly used AI algorithms include traditional machine learning methods and deep learning methods, among which Reinforcement Learning (RL) is most popular as the mechanism of dynamically allocating resources in the cloud.

In this report, we present a survey on the auto-scaling mechanisms of resources in serverful and serverless computing environments. We will focus on AI-based resource provisioning methods, and divide those methods into two categories, methods based on Reinforcement Learning and others, to provide a complete review while maintaining the importance of RL in this field. To enlarge the range of our topics, we also present some research in serverless clouds, which is expected to become the default computing paradigm, largely replacing serverful computing and thereby bringing closure to the Client-Server Era [2].

The rest of the report is organized as follows: Section 2 provides a brief introduction to the process of resource allocation in clouds and identifies some challenges. The first part of Section 3 illustrates and classifies the proposed RL-based approaches in serverful clouds, and the second part summarizes some other AI-based methods. Section 4 presents several AI-based methods in serverless clouds, and Section 5 concludes and discusses the work of this report.

2 Resource Allocation in Clouds

2.1 Process of resource allocation

Resource allocation is a classic automatic control problem, which demands a controller that dynamically tunes the type of resources and the number of resources allocated to reach certain performance goals, reflected as the SLA. Specifically, it is commonly abstracted as a MAPE (Monitoring, Analysis, Planning, and Execution) control loop [3] as shown in Fig 1.

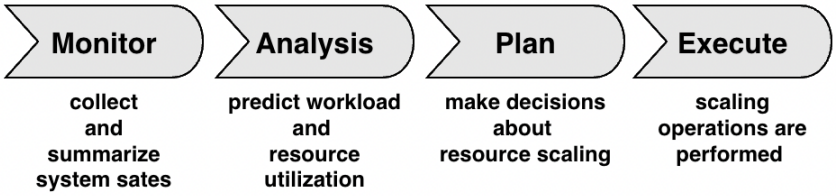

Figure 1: The MAPE control loop

A monitoring system is necessary for resource allocation, which collects and summarizes the states of applications and the infrastructure in runtime environments. Status information about current systems and pre-defined SLA requirements between the provider and the client will then be utilized by service providers to analyze and make plans for resource scaling.

Analysis and planning are stages where resource scaling mechanisms mainly participate. During the analysis stage, resource allocation systems will collect and process the performance metrics from the monitoring system and analyze current resource utilization. Some resource scaling approaches

will predict future resource demands of applications, which are called proactive methods. Methods without the prediction stage are called reactive ones, which just respond to the current status of systems and can not handle sudden bursts without performing any anticipation. Proactive methods are usually more complicated and more stable than reactive methods by making enough anticipation. In this report, most of the proposed works are about proactive methods.

In the planning stage, auto-scalers need to make decisions about whether to decrease or increase the number of resources and what type of resources need to be scaled by considering both current system utilization and SLA requirements. There are two types of resource scaling mechanisms: horizontal scaling and vertical scaling. In horizontal scaling, the system is scaled out or in by adjusting the number of assigned machines in the pool of resources to an application, such as the WMs of any type in normal clouds or the function instances in serverless clouds. On the other hand, the number of machines will not be changed in vertical scaling. The system is scaled up or in by adjusting the capabilities of individual machines, such as changing the CPU, memory, I/O of existing VMs or containers.

Execution of resource provisioning is a straightforward phase after determining the actual plans of scaling actions and is implemented by cloud providers. Details of executing are hidden from clients and beyond the scope of this paper. There is one thing to remember, that the time between the resource being requested and the resource being available for work may cause delay and influence the SLA.

2.2 Challenges in resource allocation

Runtime environments in clouds are complicated and user demands in clouds have high variance, which makes the resource allocation more difficult and problematic. Several challenges need to be considered and solved in the allocation of system resources:

Cold start delay Cold start delay is caused by the time between requesting resources and resources being available. When new resources are allocated to the system, no matter new machines or added capabilities of the machine, time is needed for the system to be reconfigured and for data to be relocated. The startup time of VMs in normal clouds is usually in orders of fractions of a second, and the latency of containers in serverless executions is largely dependent on each function's runtime dependencies and at times could grow to be even a few seconds [4], [5].

Resource efficiency Billing in cloud computing is based on a pay-as-you-go model, where customers only need to pay for resources they actually consume, thus users want the efficiency of resource utilization to be high enough to decrease the cost. In serverless clouds, the charge is only for the resources provisioned during the execution time of applications, which is usually in the order of milliseconds. Therefore, the allocation problem on serverless clouds is more complicated and needs to be paid special attention in serverless clouds.

State management The runtime environment of clouds is complicated and is measured by numerous metrics, including resource usage metrics, QoS metrics, and metrics related to configuration aspects or the status of the deployed containers or VMs [6]. Those metrics demonstrate the status of applications and resource utilization of the infrastructure, which provide information for auto-scalers to analyze and plan the resource provisioning. However, the state space over a while can be extremely large, slowing down the speed of analysis, thus an efficient definition of metrics is important when designing the algorithm.

Diverse applications Types of applications on clouds are diverse, including compute-intensive applications, such as scientific workload applications or deep learning applications, data-intensive

applications, such as data analysis applications, and cloud services, like user requests response applications. Each type of application has different expectations and SLAs, and it's hard to design just one kind of resource allocation algorithm to meet the demands of all types of applications.

Those challenges increase the difficulty of implementing an efficient auto-scaler and there are three main problems when scaling the resources: under-provisioning, over-provisioning and oscillation. When the system is over-provisioning, there are more resources to use than those needed to satisfy the SLA, which causes unnecessary costs for the client. On the contrary, in the under-provisioning case, the system does not have enough resources to process all sub-tasks or user requests, and the SLA requirement might be violated. Oscillation is where the system changes too quickly than the reaction speed of auto-scalers, thus the system bounces from under-provisioning and over-provisioning.

As stated before, resource management schemes can be classified as reactive and proactive categories which in the first case, when the workload increases/decreases to a predefined specific threshold, resource management will be conducted [7]. Reactive methods lack anticipations about future resource demands and usually cause over-provisioning when trying to meet the SLA. Although proactive methods can make predictions about the status of the system, they still cannot make completely accurate predictions and suffers from the three problems above. In this review paper, we will introduce some previous works on resource allocation and analyze what aspects of challenges they strive to solve.

3 AI based Resource allocation in cloud computing

There are four main different types of auto-scaling mechanisms in cloud computing, including static and dynamic ones, which can be categorized into five classes: static threshold-based rules, control theory, reinforcement learning, queuing theory and time series analysis [1]. In this paper, we mainly focus on dynamic AI-based mechanisms, which can learn allocation rules from resource utilization of runtime systems and adjust themselves without the intervention of system administrators. We first introduce, compare and contrast previous works about the Reinforcement Learning (RL) method, which is the most popular and powerful AI-based method on dynamic resource scaling. Then, we introduce some other AI methods, such as Support Vector Machine (SVM), Markov Model or Neural Networks (NN), to make our work more complete.

3.1 Reinforcement learning methods

Reinforcement Learning is dynamic planning algorithm, which can learn from states of systems in the runtime and improve itself according to the received rewards after taking actions. These properties make it suitable to plan resource provisioning in complicated and dynamic cloud computing environments. In this section, some basics about the RL algorithm are given at first to help understand, then some proposed works about RL are summarized and compared.

3.1.1 Basics of Reinforcement Learning

Reinforcement learning focuses on learning through direct interaction between an agent and its environment [8]. The agent in resource scaling learns the best action to take based on the current status of the application and infrastructure. Results of actions are measured by rewards received after executing those actions, which need to be as high as possible when the agent or auto-scaler decides a plan. Then the agent improves its rules about the action to take at different states based

the reward received in the previous steps in an iterative manner. The interaction between the agent and the environment is shown in Fig 2.

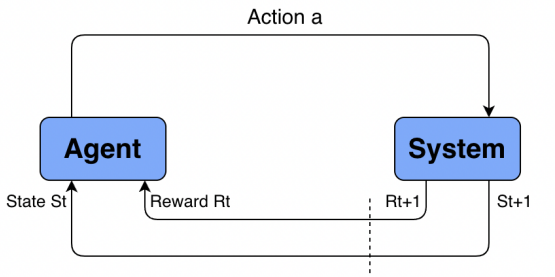

Figure 2: The architecture of RL algorithm

RL algorithms in cloud resource provisioning are commonly divided into three types, containing Q-learning, SARSA and Deep Q-learning. In Q-learning, the policy of mapping all system states to their best actions are represented by the Q(s, a) value function and stored in a look-up table. After taking action a, state s becomes to state s', and its Q-value Q(s, a) is updated by considering both the maximum reward of Q(s', a') and the direct reward by taking action a. In contrast to Q-learning, SARSA is a more conservative strategy. When updating the Q-value, SARSA has already planned the action for the future and is more sensitive to mistakes. However, Q-learning selects the direction that maximizes Q-value each time it is updated and then re-selects the action in the next state. In the Deep learning case, the table with Q values is replaced by a deep neural network to represent the action-value function [6].

3.1.2 Review of previous works

Before applying RL methods in practice, three basic components have to be identified: the state set, the action set and the reward metric. The first two elements are highly related to the scaling mechanisms, and states and actions defined in horizontal scaling and vertical scaling are of big difference. In horizontal scaling, states like input workload and number of VMs are mostly defined and actions are like removing a VM, adding a VM or doing nothing. On the other hand, states in vertical scaling are usually about the capabilities of resources in each machine, such as CPU and memory. Actions are like adding or decreasing the number of resources in each machine or doing nothing.

Agent in RL learns from the feedback of previous states and rewards of the system, and then it tries to scale up or down the system by choosing the right action [9]. Some works integrate RL algorithms into their reactive models and respond to the system when received signals, such as the work by Barrett et al [10]. This work creates an agent for each VM, which maintains its own lookup table. These agents are trained in parallel, which is computationally expensive and inefficient. Besides, the reactive model is unable to handle sudden bursts of tasks, thus it suffers from over-provisioning when it tries to meet SLA requirements but causes a waste of money.

Other works employ reinforcement learning in a proactive way, which takes action in a fixed period of time or predicts the need for scaling resources and makes scaling decisions when the need

is detected. One SARSA-based reinforcement learning method, RLPAS, is presented by Benifa et al [11]. The purpose of this work is to dynamically provision the resources to maximize resource utilization while reducing response time and increasing throughput. The states of this method include the number of user requests and resource utilization statistics, throughput and response time. The reward for state-action pairs is defined by the trade-off between application performance and resource utilization. Researchers compared this method with one basic Q-learning algorithm and one basic SARSA algorithm from reference works. After the experiment, they found that this method can converge in a shorter time and performs better in the throughput and resource utilization than another two baseline methods.

Asghari et al [12] proposed a resource management framework, RMFW, including scheduling management, resource provisioning management, and VM management. This framework aims to find a satisfactory compensation between several objectives, which are reducing user costs, maximizing utilization of resources, and reducing job latency. It consists of multiple cooperative reinforcement learning agents, which are assigned to different resources and are responsible for selecting the most suitable task from all ready tasks. It uses the Markov game algorithm to integrate multiple learning agents and calculate an overall reward for all agents. States in each learning agent are defined as different resource utilization modes, and agents will take actions by selecting the next ready tasks to execute. The reward is based on the association between resource utilization and execution time. This Q-learning based reinforcement learning algorithm is tested in a simulation environment with Workflowsim [13] framework, and is compared with five previous works.

Bitsakos et al [14] present the DERP (Deep Elastic Resource Provisioning) framework with three different agents, including one simple Deep Q-learning agent and two complicated Deep Q-learning agents. This framework was designed to reduce the complexity of a large size state space to increase the speed of converging. It monitors the cloud system by several metrics that describe the resource utilization, the number of user requests, and the number of clusters with different memory states. In each state, the learning agent will choose one action from increasing one cluster, decreasing one cluster, or doing nothing and will receive rewards to update the NN, which are determined by the throughput and latency of jobs and the number of VMs. To evaluate the performance of their framework, researchers deploy this method in Okeanos cloud services and test it in a real-world environment. Experiment results show that it converges rapidly to the runtime environment and manages to achieve the optimal behavior at most times.

Arani et al [15] aims to deal with fluctuating demands and proposes a stable strategy. To minimize total cost while maximizing resource utilization, it defines three discrete states: normal-utilization, under-utilization, and over-utilization, inferred from the CPU utilization. The Q-learning based scaling agent can choose to scale in, scale out or do nothing and receives rewards from one predefined table, storing the transition matrix between states and actions. The experiment is simulated in the Cloudsim toolkit [16] with two real-world workload traces and compared to three baseline strategies. Researchers define five performance metrics to evaluate the method: resource utilization, the number of VMs, SLA violation, total resource cost, and profit. Similar to the work by Arani et al [15], this work in Arabnejad et al [17] also aims at handling various workload traffics in the runtime environments. It combines two standard RL strategies, i.e. Q-learning and SARSA, with fuzzy logic. States about the system in this work are defined in a fuzzy manner, including nine states about the workload and response time. It proposes two methods for reinforcement learning, fuzzy SARSA learning (FSL) and fuzzy Q-learning (FQL) [17]. Experiments on an open-source IaaS platform, OpenStack[1] with different workloads of user HTTP requests demonstrate the effectiveness

[1] https://www.openstack.org

and improvement of this method.

Besides resource utilization, some works also consider power consumption while maximizing the throughput of jobs. Liu et al [18] proposes a hierarchical framework to solve resource provisioning in the global tier and power management in the local tier at the same time. It also combines the offline pre-trained DNN with an online Q-learning agent to make accurate scaling decisions more rapidly and decrease the dimensions of state-action pairs. States in the global resource scaling tier consist of utilization requirement and job duration of applications and utilization level of the infrastructure. The learning agent will obtain rewards by deciding the index of the server where the job is assigned to and rewards are measured by the negative weighted sum of the total energy cost, the number of VMs, and the reliability penalty. This method was compared with a standard round-robin VM allocation policy, and experiment results showed that it saves the power consumption than the reference work and achieves similar average latency.

Another different and novel method was proposed by Wei et al [19], which helps providers in SaaS services to find the optimal policy of renting IaaS infrastructures to meet their clients' demand while minimizing renting expenses and maximizing resource utilization. It considers the different types of VMs(on-demand and reserved) and corresponding different pricing models. The state set of this Q-learning method includes all possible states that are related to a SaaS provider: the workload of applications, the number of VMs of each type, and a time-stamp within the workload period. The action set defines how many VMs of each type are to be assigned for the next execution round, and rewards describe the total earnings SaaS providers can make via taking action a plus the performance of the applications after action a. The experiment is designed to assess three kinds of market situations: on-demand, reserved, and hybrid. In their experiments, the reward values reach stability after several rounds and their method attains a satisfactory result.

3.2 Other methods

In addition to reinforcement learning based scaling mechanisms, other methods based on basic machine learning algorithms or statistical models are also extensively used in resource allocation. These approaches are trained on pre-defined datasets and can be improved in real running environments to save training time. This section introduce some previous works on resource scaling using traditional ML algorithms, including Support Vector Machine (SVM), Markov Model, or Neural Networks (NN).

To prove the hypothesis that the prediction accuracy of auto-scaling systems can be increased by choosing an appropriate time-series prediction algorithm based on the performance pattern over time [20], this work by Nikravesh et al in [20] conducts experiments under different performance patterns to compare the accuracy of time-series based models on predicting future workloads. This paper adopts SVM and Neural Network as their prediction model, considering that they are the most effective prediction algorithms to predict future system characteristics [21]. Finding suitable time window sizes for these time-series models to include enough correlation between different periods but avoid over-fitting at the same time is non-trivial work. Therefore, these two models are pre-trained on the simulated workloads created by the TPC-W workload generator and are deployed and tested against the testing dataset to fit the data and find the best hyperparameter. Researchers utilize traditional evaluation metrics of regression problems, such as Mean Absolute Percentage Error and Root Mean Square Error, to measure the performance of these models. Experimental results show that SVM and NN outperform each other in different workload patterns, thus it's useful to the select most suitable prediction model according to the performance pattern.

Ajila et al [22] presents a machine learning classifier to improve QoS while reducing user cost as much as possible by adjusting the number of Virtual Network Functions (VNFs) in the cloud. This classifier learns seasonal patterns of traffic workloads from previous experience and considers the underlying property of virtualization technology, such as the cold start time of virtual machines. By defining about 20 features containing the information about the time of that day, measured traffic at that time, and traffic change during the time interval, this classifier can learn temporal patterns of traffic workloads. With the input of features describing the current status of the system, the classifier will output the number of VNFs needed to handle current traffic. This classification model is based on several popular ML algorithms, including decision-tree, Bayesian Network and Random Forest, and experiment results demonstrate the promising accuracy of this classifier.

Simic et al [23] proposes a framework for applications about optimization problems. The main component of this framework is an intelligent decision support engine for users to obtain an optimization of the execution time of applications and resource cost. This decision engine is based on artificial neural networks (ANN) and metaheuristics, which can provide evaluation results about the performance of the infrastructure in the aspects of job execution time and the cost of resources. Through the decision support engine, this auto-scaling framework is able to determine the optimal strategy to make a compensation between execution time and resource cost. There are two ANNs in this framework, one is for estimating the time of the whole decision-making process and another is for estimating the total engagement time of computing resources. Combining these two predicted outputs, the decision-making engine will recommend optimal parameters for the framework to minimize the execution time for the application. Experiments about optimization applications in real-world simulations show that this decision support engine can save the infrastructure cost with the same job execution time.

Apart from traditional machine learning based scaling approaches, some works utilize Markov models to allocate resources. Nikravesh et al [24] proposes an auto-scaling system based on the Hidden Markov Model (HMM) to minimize resource cost while maximizing the performance of applications. Hidden Markov Models, as improved versions of Markov models, can model time-series data and capture patterns in data when states are not visible to make predictions. To thoroughly describe system conditions, researchers define a metric set containing multiple performance metrics about the application and the infrastructure. The historical data for experiments are created by the TPC-W benchmark generator, 60% of which are for training and the rest are for testing. According to the experiment result, their HMM framework can make effective provisioning decisions approximately 97% of the time.

Based on the one-layer HMM published in [24], Runsewe et al [25] presents a Layered Multi-dimensional Hidden Markov Model (LMD-HMM) to analyze resource allocation problems for big data streaming applications to broaden the range of types of applications. The changing data streams in big data streaming applications are highly unpredictable, thus those kinds of applications are easy to suffer from resource oscillation. However, big data streaming applications have strict requirements for job latency, which increases the difficulty of provisioning resources. To overcome those challenges, the LMD-HMM in this work proposes three cooperative layers to make decisions, including a data ingestion layer, a processing layer, and a storage layer with a scaling decision-making model. The hidden workload-related states defined in this model, including the number of streams and the arrival rate of workloads, help the model to capture the distribution of application workloads and make predictions according to the probability distribution.

4 AI-based Resource Allocation in Serverless Clouds

As a new emerging computing paradigm in cloud computing, serverless cloud computing does obtain a lot of attention and an increasing number of research works arise in resource allocation of serverless clouds. However, the applicability of RL-based technology to optimize auto-scaling capabilities in serverless environments has not been adequately investigated [26]. Therefore, this section will introduce some AI-based auto-scaling mechanisms, especially those based on reinforcement learning, to explore the development of reinforcement learning in resource provisioning of serverless cloud computing.

4.1 AI-based methods

Resource provisioning for serverless functions is fine-grained spatially (i.e., small resource volumes) and temporally (i.e., short available time) [27], which means that resource requests in serverless environments are shorter and have higher variance than in serverful clouds. Those properties increase the difficulties for serverless service providers to achieve a stable resource scaling mechanism while meeting the SLA. This section present some AI-based strategies from previous works on resource scaling in serverless clouds.

Bhattacharjee et al [28] implements an intelligent resource scaling agent to manage the compute resources in their Deep Learning Prediction Services system in serverless clouds. This scaling agent is based on the deep learning algorithm and is able to predict future resource demands with workload patterns learned from historical data. Resource allocation in this work is mainly by horizontal scaling, and when the agent detects the need of scaling up/down resources, i.e. VMs in this context, it will increase/decrease the number of VMs. One novel idea of this work is that when the deep learning agent detects the over-provisioning of resources, it will allocate resources to the low-priority batch jobs by vertical scaling. Through the implementation of both horizontal scaling and vertical scaling, resource efficiency is improved.

One container-based resource management system for serverless computing is proposed in Somma et al [29], which combines the admission control function and resource provisioning function. This self-adaptive system minimizes the application's response time and resource cost while maximizing the application's throughput through reinforcement learning. The admission controller limits the access to internal containers and the auto-scaler is responsible for adding or removing containers. The auto-scaler defines states as a triplet set, describing resource utilization in the previous and current loop and the number of containers, and adjusts the resource by adding or removing one container or doing nothing. The reward function of this RL agent is defined as the blocking rate of applications and the number of provisioned containers. Experimental results of comparison with the open-source container manager Kubernetes prove the efficiency of this resource management system.

Some workload-based or request-based auto-scalers start the scaling action when a predefined maximum number of workloads or requests that the system can process in parallel is reached. Therefore, a suitable maximum number of workloads or the maximum concurrency level for different types of applications is necessary to meet the SLA while increasing resource utilization. Schuler et al [26] proposed one solution to this problem, which implements a Q-learning based agent for automatically scaling resources in serverless clouds. States in this RL agent contain information about resource utilization and the concurrency level. This agent allocates resources by adjusting the concurrency level via decreasing or increasing 20 requests every time or doing nothing.

The auto-scaling method proposed by Zafeiropoulos et al [6] considers not only the workload

properties but also throughput, latency, and other SLA requirements. In order to examine the provisioning performance achieved by their approach in different settings for different types of applications, they proposed two kinds of state sets, continuous state set and discrete state set. RL algorithms are implemented on the basis of Q-learning and DynaQ+ respectively and are tested in each environment. After a series of comparative experiments, the results show that there is a minor difference in scaling performance between continuous state set and discrete state set. Considering that the discrete state set can save training time and storage, it can be a default choice in many situations.

5 Classification of Previous Works

This section gives a deeper comparative analysis of the reviewed works according to defined taxonomies. We propose three metrics that describe the QoS objectives accomplished in each work, the types of conducted applications, and the types of adopted algorithms respectively.

Table 1: Allocation Methods in Clouds

Work	QoS goal	Application	H/V	R/P	Algorithms	Environments
Barrett2013 [9]	Performance, Cost	Cloud services	H	R	Q-learning	Serverful
Benifa2019 [11]	Utiliz., Response Rime, Throughput	Workflow	H	P	SARSA	Serverful
Asghari2020 [12]	SLA, Cost	Workflow	H	P	Q-learning	Serverful
Bitsakos2018 [14]	Utiliz., Cost	Cloud services	H/V	P	Deep Q-learning	Serverful
Arani2018 [15]	Utiliz., Cost, SLA, Profit	Workflow	H	P	Q-learning	Serverful
Arabnejad2017 [17]	Response Time, Cost	Workflow	H	P	Fuzzy Q-learning, Fuzzy SARSA	Serverful
Liu2017 [18]	Latency, Energy consumption, Throughput	Workflow	H	P	Deep Q-learning	Serverful
Wei2019 [19]	Utiliz., Cost	Cloud services	H	P	Q-learning	Serverful
Nikravesh2015 [20]	Prediction accuracy	Workflow	H	P	SVM, NN	Serverful
Rahman2018 [22]	Cost, SLA	Workflow	H	P	Machine Learning	Serverful
Simic2019 [23]	Execution Time, Cost	Workflow	H	P	ANN	Serverful
Nikravesh2014 [24]	Performance, Cost	Workflow	H	P	HMM	
Nikravesh2014 [24]	Latency, Cost	Independent Tasks	H	P	HMM	Serverful
Anirban2019 [28]	Utiliz., Cost	Workflow	H/V	P	Deep Learning	Serverless
Somma2020 [29]	Response Time, Throughput, Cost	Cloud Services	H	P	Q-Learning	Serverless
Schuler2021 [26]	Performance, Cost	Workflow	H	P	Q-learning	Serverless
Anastasios2022 [6]	Throughput, Latency, SLA	Workflow	H	P	Q-learning, DynaQ+	Serverless

QoS objectives in resource allocation contain application performance, such as latency, throughput and response time, and resource utilization, such as user cost, resource efficiency and energy efficiency. Applications in the cloud are divided into three types according to the criterion adopted by Garí et al in [30], including workflows, independent tasks, and cloud services. Workflow applications consist of a set of dependent and readily-executable tasks, while independent tasks refer to a series of logically-related but independent tasks. Cloud services handle user requests or other types of dynamic workloads. The RL algorithms are categorized into Q-learning, SARSA and Deep Q-learning.

The comparative result of proposed methods in serverful and serverless clouds is presented in Table 1, where H/V represents horizontal or vertical scaling, R/P represents reactive or proactive models and Utiliz. represents resource utilization.

6 Conclusion

In this report, we surveyed research into resource allocation approaches based on Artificial Intelligence in serverful and serverless clouds. We first briefly introduce the MAPE process of resource provisioning and identify some challenges in scaling resources. Then we summarize and compare some proposed research works in AI-based allocation methods in serverful and serverless clouds, according to the aspects of QoS objectives, types of applications, scaling mechanisms, and adopted algorithms.

The main focus of our work is on Reinforcement Learning approaches, because of their effectiveness and popularity. RL-based scaling strategies can be divided into three categories: 1) Q-learning; 2) SARSA; 3) Deep Q-learning. One of the important findings of this research is that most published works on serverful clouds are based on Q-learning or SARSA, but there are an increasing number of papers adopt Deep Q-learning for resource scaling in recent times. Some papers also use other techniques in AI, such as Deep Neural Network or Random Forest. The second finding of our work is that Reinforcement Learning is not commonly used in serverless clouds, and those papers we discussed in Section 4 are mainly based on Q-learning or improved Q-learning.

References

[1] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of grid computing*, 12(4):559–592, 2014.

[2] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*, 2019.

[3] Chenhao Qu, Rodrigo N. Calheiros, and Rajkumar Buyya. Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Comput. Surv.*, 51(4), jul 2018.

[4] Amoghavarsha Suresh, Gagan Somashekar, Anandh Varadarajan, Veerendra Ramesh Kakarla, Hima Upadhyay, and Anshul Gandhi. Ensure: Efficient scheduling and autonomous resource management in serverless environments. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 1–10. IEEE, 2020.

[5] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking behind the curtains of serverless platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 133–146, 2018.

[6] Anastasios Zafeiropoulos, Eleni Fotopoulou, Nikos Filinis, and Symeon Papavassiliou. Reinforcement learning-assisted autoscaling mechanisms for serverless computing platforms. *Simulation Modelling Practice and Theory*, 116:102461, 2022.

[7] Emanuel Ferreira Coutinho, Flávio Rubens de Carvalho Sousa, Paulo Antonio Leal Rego, Danielo Gonçalves Gomes, and José Neuman de Souza. Elasticity in cloud computing: a survey. *annals of telecommunications-annales des télécommunications*, 70(7):289–309, 2015.

[8] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

[9] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Elasticity in cloud computing: State of the art and research challenges. *IEEE Transactions on Services Computing*, 11(2):430–447, 2018.

[10] Enda Barrett, Enda Howley, and Jim Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and computation: practice and experience*, 25(12):1656–1674, 2013.

[11] JV Bibal Benifa and D Dejey. Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment. *Mobile Networks and Applications*, 24(4):1348–1363, 2019.

[12] Ali Asghari, Mohammad Karim Sohrabi, and Farzin Yaghmaee. A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents. *Computer Networks*, 179:107340, 2020.

[13] Weiwei Chen and Ewa Deelman. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *2012 IEEE 8th international conference on E-science*, pages 1–8. IEEE, 2012.

[14] Constantinos Bitsakos, Ioannis Konstantinou, and Nectarios Koziris. Derp: A deep reinforcement learning cloud system for elastic resource provisioning. In *2018 IEEE international conference on cloud computing technology and science (CloudCom)*, pages 21–29. IEEE, 2018.

[15] Mostafa Ghobaei-Arani, Sam Jabbehdari, and Mohammad Ali Pourmina. An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. *Future Generation Computer Systems*, 78:191–210, 2018.

[16] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[17] Hamid Arabnejad, Claus Pahl, Pooyan Jamshidi, and Giovani Estrada. A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In *2017 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID)*, pages 64–73. IEEE, 2017.

[18] Ning Liu, Zhe Li, Jielong Xu, Zhiyuan Xu, Sheng Lin, Qinru Qiu, Jian Tang, and Yanzhi Wang. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pages 372–382. IEEE, 2017.

[19] Yi Wei, Daniel Kudenko, Shijun Liu, Li Pan, Lei Wu, and Xiangxu Meng. A reinforcement learning based auto-scaling approach for saas providers in dynamic cloud environment. *Mathematical Problems in Engineering*, 2019, 2019.

[20] Ali Yadavar Nikravesh, Samuel A Ajila, and Chung-Horng Lung. Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 35–45. IEEE, 2015.

[21] Samuel A Ajila and Akindele A Bankole. Cloud client prediction models using machine learning techniques. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 134–142. IEEE, 2013.

[22] Sabidur Rahman, Tanjila Ahmed, Minh Huynh, Massimo Tornatore, and Biswanath Mukherjee. Auto-scaling vnfs using machine learning to improve qos and reduce cost. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[23] Visnja Simic, Boban Stojanovic, and Milos Ivanovic. Optimizing the performance of optimization in the cloud environment–an intelligent auto-scaling approach. *Future Generation Computer Systems*, 101:909–920, 2019.

[24] Ali Yadavar Nikravesh, Samuel A Ajila, and Chung-Horng Lung. Cloud resource auto-scaling system based on hidden markov model (hmm). In *2014 IEEE International Conference on Semantic Computing*, pages 124–127. IEEE, 2014.

[25] Olubisi Runsewe and Nancy Samaan. Cloud resource scaling for big data streaming applications using a layered multi-dimensional hidden markov model. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 848–857. IEEE, 2017.

[26] Lucia Schuler, Somaya Jamil, and Niklas Kühl. Ai-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 804–811. IEEE, 2021.

[27] Hanfei Yu, Hao Wang, Jian Li, and Seung-Jong Park. Harvesting idle resources in serverless computing via reinforcement learning. *arXiv preprint arXiv:2108.12717*, 2021.

[28] Anirban Bhattacharjee, Ajay Dev Chhokra, Zhuangwei Kang, Hongyang Sun, Aniruddha Gokhale, and Gabor Karsai. Barista: Efficient and scalable serverless serving system for deep learning prediction services. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 23–33. IEEE, 2019.

[29] Gaetano Somma, Constantine Ayimba, Paolo Casari, Simon Pietro Romano, and Vincenzo Mancuso. When less is more: Core-restricted container provisioning for serverless computing. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1153–1159. IEEE, 2020.

[30] Yisel Garí, David A Monge, and Cristian Mateos. A q-learning approach for the autoscaling of scientific workflows in the cloud. *Future Generation Computer Systems*, 127:168–180, 2022.